Survey: Attacks on Hash functions

Andrea VISCONTI and Sergio POLESE

Department of Computer Science
Università degli Studi di Milano

Overview

1. Properties of Hash Functions
   - Collision free
   - Pre-image resistant
   - Second pre-image resistant

2. Attacks
   - Collision attacks
   - Pre-image attacks
   - Second pre-image attacks

Properties of Hash Functions

Hash Functions

What are hash functions?

Hash Functions are cryptographic functions. They

- can be **efficiently computed**;
- input **any string** of any size;
- provide a **fixed size output**.

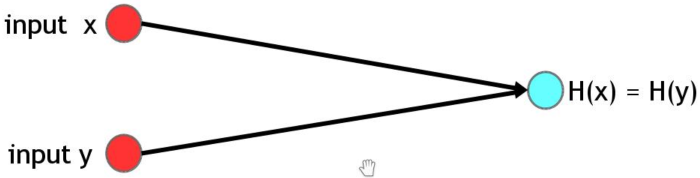
Some examples of hash functions:

- RIPEMD160 –> a message digest (MD) of 160 bits;
- SHA1 –> a message digest of 160 bits;
- SHA-256 –> a message digest of 256 bits;
- MD5 –> a message digest of 128 bits;
- SHA-3 –> . . .

Hash Functions

Properties...

1. **Collision-free**



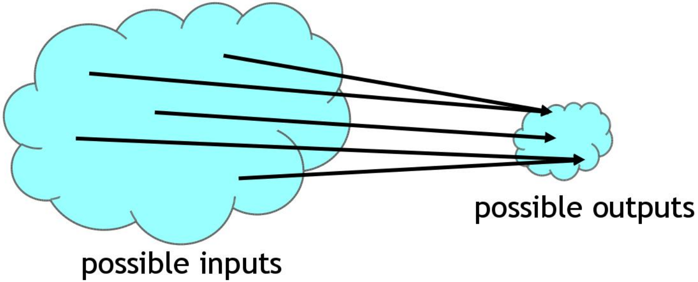
Nobody is able to find two strings x and y s.t. $x \neq y$ and $H(x) = H(y)$

Hash Functions

Collision do exist!

Indeed,

- we input any string of any size — say n, this means 2^n;
- we provide a fixed size output — for example 256 bits, this means 2^{256};



But are you able to find them?

Hash Functions

I can suggest you an algorithm...

If you compute the hash of 2^{130} strings, you have more than 99% chance that two inputs collide (at least two!!).

Unfortunately these **collisions are not findable by regular users** using regular computers because this process takes a very very long time.

This number is astronomical!

The **probability** to find a collision is **negligible**.

Hash Functions

We have understood that

- **no hash function** has been **proven to be collision free**;
- it is very hard to find a collision.

So, **we choose to believe** that hash functions are collision free.

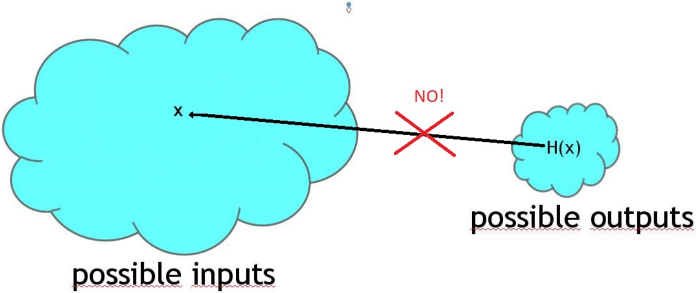
Therefore we assume that if $H(x) = H(y)$ then $x = y$.
This means that

- we do not store x and y (that can be huge) but only their hashes (usually very small);
- we are able to compare data using only a bunch of bits.

Hash Functions

Properties...

2. **Preimage resistant**



Given $H(x)$, it is computationally infeasible to find x.

Hash Functions

Preimage resistance refers to the hash function's ability to be non-reversible: **one-way function**.

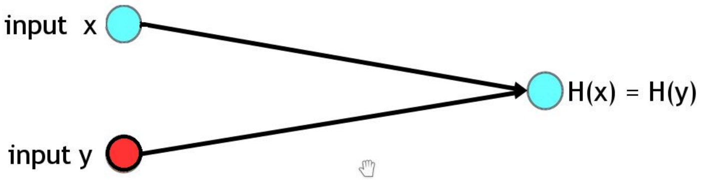
In particular, this property suggest us that

- there is no value of x which is particularly likely;
- attackers have to try all possible input values to find x. Again, this number is astronomical!
- we can hide x using $H(x)$.

Hash Functions

Properties...

3. **Second preimage resistant**



Given x, it is computationally infeasible to find y s.t. $H(x) = H(y)$.

These three properties ensure that it is hard to cheat.

Merkle-Damgard construction

Several hash functions (e.g. MD5, SHA-1, SHA-2) are based on the so called **Merkle-Damgard construction**:

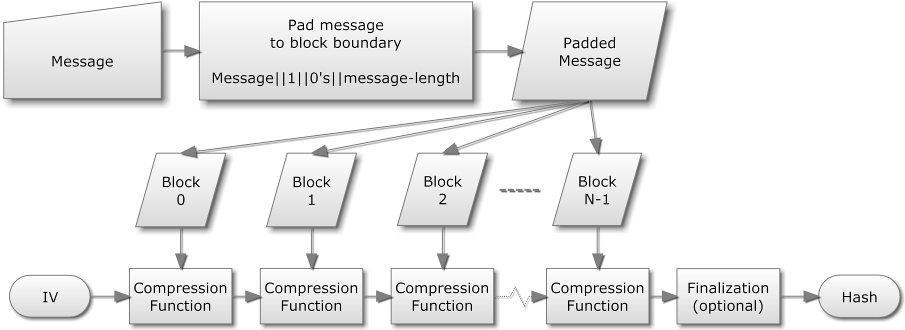


Figura: Merkle-Damgard construction (Marc Stevens, "Attacks on hash functions and applications", Leiden University, June 2012)

Attacks

Brute-force attacks

Brute-force attacks don't depend on the specific algorithms but only on the bit length of the hash value.
The table below summarizes the computational cost of the brute-force attacks on a hash function with a *m*-bit message digest:

Collision resistance	$2^{(m/2)}$
Preimage resistance	2^m
Second preimage resistance	2^m

Cryptanalysis

Cryptanalysis of hash functions studies carefully the internal structure of the compression function trying to find possible weaknesses to take advantage of.
These are the computational costs of the attacks known today to some hash functions as regards collision and pre-image resistance:

	$MD4$	$MD5$	$RIPEMD$	$SHA-1$
Collision resistance	2^2	2^{24}	2^{18}	2^{63}
Preimage resistance	2^{95}	2^{123}	2^{113} (35 steps)	$2^{159.3}$ (62 steps)

Differential cryptanalysis

The most successful cryptanalysis on hash functions is the one against collision resistance. It is mostly made up of the so-called **differential cryptanalysis**:

- Publicly known since the published attacks on block ciphers and cipher-based hash functions by Biham and Shamir in 1980s;
- Method which compares the output of the function applied to messages that differ only for a small number of bits;
- Successfully applied to hash functions such as MD4, MD5, SHA-1 and many others;

Differential cryptanalysis for collisions

Some general steps of differential cryptanalysis applied to hash functions in order to find collisions:

- Find a **differential path**, i.e. a group of vectors which describe precisely how differences between the two input pairs propagate through the compression function;
- Impose some **conditions** on the working states letting the attacker control the output of the boolean function and the modular addiction in a certain step in order to maximise the success probability of the differential path;
- **Apply the selected message differences** to a bunch of messages M to obtain a collision. The probability obtained in the previous step gives the expected number of messages we have to test until we possibly find a collision, thus the computational cost of the attack.

Selecting a good differential path

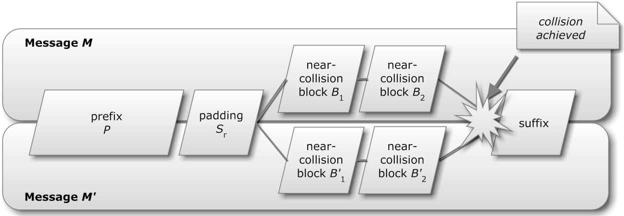
Some general methods to find the best differential path are the following:

- Starting from a **disturbance vector** (i.e. a vector that tells which bits of the original message M are modified), a differential path is obtained trying to correct the errors in order to find a local collision for a **linearized** version of the compression function (e.g. SHA-1);
- Exploiting the **non-linear** parts of the compression function, which usually are the boolean functions and the modular addiction. This gives more flexibility for the differential path.

Tricks to improve the search

To improve the search it is possible to use some useful tricks:

- Use an **identical prefix** P so that the messages M and M' are of the form $M = P||B$ and $M' = P||B'$. This means finding a collision which starts from different initial values;
- Use **multi-block messages**, that is $M = M_1|| \dots ||M_n$ and $M' = M_1'|| \dots ||M_n'$, so that a full collision is achieved only after the last block while in the middle ones near collisions are found. This is particularly useful to reduce the complexity of the attack when a single-block collision is more complex.



Deriving the sufficient conditions

While the conditions in the first round (usually first 16 steps) are simple to satisfy, it is more difficult to satisfy the ones for later steps so that some advanced techniques have to be used:

- **Neutral bits**: given two messages M_1 and M_2 that satisfy the differential path up to step t there are some "neutral bits" that, if flipped, generate M_1' and M_2' which also satisfy the differential path;
- **Tunnels**: changing some bits in a certain working state of the first round causes changes in the second round only after some specific step l;
- **Boomerangs**: briefly a small set of bits that together form a local collision;

Example : SHA-1

The step function of SHA-1 is the following:

$$Q_{t+1} = Q_t^{\lll 5} + F_t + Q_{t-4}^{\lll 30} + W_t + K_t, \tag{1}$$

where

$$F_t = f_t(Q_{t-1}, Q_{t-2}^{\lll 30}, Q_{t-3}^{\lll 30}), \tag{2}$$

K_t is a constant and W_t is obtained from the expansion of the message words with a specific algorithm. So the linearized version is the following:

$$Q_{t+1} = Q_t^{\lll 5} \oplus Q_{t-1} \oplus Q_{t-2}^{\lll 30} \oplus Q_{t-3}^{\lll 30} \oplus Q_{t-4}^{\lll 30} \oplus W_t \oplus K_t, \tag{3}$$

Results on collision attacks

A combination of all these techniques gave collision attacks to several hash functions:

- An attack to MD4 costs just 2^2 computations. (Naito, **2006**)
- In **2013**, an attack to MD5 was proposed with complexity 2^{18} by Xie, Liu and Feng;
- RIPEMD attack of **2005** by Wang et al. has a complexity of 2^{18}. RIPEMD-160 is much more difficult to brake: the best attack is still due to Wang et al. and its time complexity is $2^{74.3}$ for a reduced version to 34 rounds.
- The first SHA-1 collision was found in **2017** by Stevens et al. The computational cost of the attack is of $2^{63.1}$.

First collision of SHA-1

$M_1^{(1)} =$ 7f46dc93 a6b67e01 3b029aaa 1db2560b 45ca67d6 88c7f84b
8c4c791f e02b3df6 14f86db1 690901c5 6b45c153 0afedfb7
6038e972 722fe7ad 728f0e49 04e046c2

$M_2^{(1)} =$ 30570fe9 d41398ab e12ef5bc 942be335 42a4802d 98b5d70f
2a332ec3 7fac3514 e74ddc0f 2cc1a874 cd0c7830 5a215664
61309789 606bd0bf 3f98cda8 044629a1

$M_1^{(2)} =$ 7**3**46dc9**1** **6**6b67e**1**1 **8f**029a**b6** **21**b2560**f** **f9**ca67**cc** **a**8c7f8**5**b
a84c79**03** **0c**2b3d**e2** 1**8**f86db**3** **a**90901**d**5 **df**45c1**4f** **26**fedfb**3**
dc38e9**6a** **c**22fe7**b**d 728f0e4**5** **bc**e046**d**2

$M_2^{(2)} =$ 3**c**570fe**b** **1**41398**b**b **55**2ef5a**0** **a8**2be33**1** **fe**a480**37** **b**8b5d7**1**f
0e332e**df** **93**ac35**00** e**b**4ddc0**d** **e**cc1a8**6**4 **79**0c78**2c** **76**21566**0**
dd3097**91** **d**06bd0**a**f 3f98cda**4** **bc**4629**b**1

Meet-in-the-middle attack

A diffused attack on the pre-image of hash functions is the **meet-in-the-middle attack** from Sasaki and Aoki.

- Since 2008, it was exploited for several attacks to hash functions whose message expansion is only a permutation of the message words (e.g. MD4, MD5);
- The technique used is the **"splice and cut"**: the attack target is divided in two chunks of steps so that each of them includes at least one message word that doesn't appear in the other chunk. Such words are called **"neutral words"**. Only after the pseudo pre-images are found using the meet-in-the-middle approach and then converted in pre-images;
- Partial matching and partial fixing techniques are other improvements to this powerful approach introduced by the two authors;

Pre-image attacks exploiting meet-in-the-middle attack

Here are listed some remarkable results that use the meet-in-the-middle approach successfully:

- Along with other techniques, a pre-image attack was found by Zhong and Lai in **2010** on MD4 with complexity $2^{94.98}$, much better than the brute-force attack;
- With some improvements (e.g. initial-structure technique) and adapting the method to the specific compression function, the best attack known today to MD5 has a computational cost of $2^{123.4}$ and was proposed by Sasaki and Aoki in **2009**;
- In **2009**, new types of local collisions called "one-message-word local collisions" are used to construct a MiTM attack by Wang et al. to reduced RIPEMD and then improved in 2012 to obtain a theoretical attack with complexity 2^{96}.

Pre-image attacks on SHA-1

- In **2008**, De Canniére and Rechberger proposed an attack based on "reversing the inversion problem" , a method which consists in finding an impossible expanded message that would lead to the required output and then correcting it. This resulted in an attack to reduced SHA-1 to 45 steps with complexity lower than brute-force;
- In **2009**, Sasaki and Aoki adapted their MiTM approach to SHA-1: the algorithm is not thought for a message expansion as the one of SHA-1 which involves a rotation and xor operations. Thanks to this they attacked SHA-1 reduced to 48 steps with complexity $2^{159.3}$;
- In **2012**, their work was improved by Knellwolf and Khovratovich, thanks to a differential view on the MiTM approach, who increased the number of rounds attacked to 57 with complexity $2^{158.8}$;
- Finally, in **2015**, Finally, this differential view is generalyzed to higher order differential to give a 62-rounds preimage for SHA-1 with complexity $2^{159.3}$.

Generic second pre-image attack

In **2005**, Kelsey and Schneier published a paper in which they show a generic attack to find **second pre-images** of all hash functions based on the Damgard-Merkle construction:

- For a message of 2^k blocks the cost of the attack is about $k \times 2^{n/2+1} + 2^{n-k+1}$;
- Even if it allows to find second pre-images with less complexity than the brute-force attack, a remarkable improvement can only be seen for impractically long messages: note that, if the message is 4 blocks long ($K = 2$), then the complexity is around $2^{n/2+2} + 2^{n-1}$, which is just slightly less than exhaustive search;

Thanks for your attention!

andrea.visconti@unimi.it
www.di.unimi.it/visconti

and

sergio.polese@unimi.it